

**TECNOLÓGICO NACIONAL DE MÉXICO**

***INSTITUTO TECNOLÓGICO DE TIJUANA***

**SUBDIRECCIÓN ACADÉMICA**

Departamento de Sistemas y Computación

**EXAMEN**

Carrera: **Ingeniería en Sistemas Computacionales** Período: **Agosto – Diciembre 2021**

Materia: **Estructura de Datos** Serie: **SC3C** Salón: **Classroom**

Unidad (es) a evaluar: **IV – Estructuras No Lineales.**

Tipo de examen: **Primera**

# **Oportunidad** Fecha: **8/noviembre/2021**

Catedrático: **M.C.C. Luz Elena Cortez Galván** Firma del maestro: Calificación:

Alumno: Covarrubias Rosales Luis Humberto No. Control: 20212393

# **EXAMEN 1:**

1 -> 7.- Elabore un Programa en C# que con un menú de opciones permita Crear un árbol binario de numeros de control de 8 dígitos **generados aleatoriamente** (NO se capturan), y Utilice el siguiente menú de opciones:

**MENU ARBOL NCONTROL**

1. **Inserción de Nodos.**

# **Recorrido en Preorden, Inorden, Posorden.**

1. **Búsquedas de forma Recursiva e Iterativa.**

# **Eliminación de Nodos.**

1. **Salir del Programa.**

# **EXAMEN 2:**

1. > 7.- Elabore un Programa en C# que con un menú de opciones permita Crear un árbol binario con los siguientes nombres : MICKEY (**raíz**), NEMO, MINNIE, DONALD, DAISY, GOOFY, PACO (**asignarlos**, NO se capturan), y Utilice el siguiente menú de opciones:

**MENU NOMBRES DISNEY**

* 1. **Inserción de Nodos.**

# **Recorrido en Preorden, Inorden, Posorden.**

* 1. **Búsquedas de forma Recursiva e Iterativa.**

# **Eliminación de Nodos.**

* 1. **Salir del Programa.**

**NOTAS:**

* Para los 2 exámenes en el **caso B)** ejecutar los 3 recorridos y mostrar los resultados obtenidos.
* Para los 2 exámenes en el **caso C)** capturar el dato a buscar y enviarlo como parámetro a los 2 métodos y mostrar los resultados obtenidos.
* En el **caso E)** para los 2 exámenes desplegar el mensaje de salida del programa y el resultado de calcular las complejidades temporal y espacial.

A

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Diagnostics;

namespace CovarrubiasLuis\_U4

{

class Program

{

class Arbol

{

int info;

Arbol izq, der;

public Arbol()

{

info = 0;

izq = null;

der = null;

}

public Arbol raiz = null;

~Arbol() => Console.WriteLine("Memoria del Arbol liberada");

public void insertar(int elemento)

{

int bandera = 0;

Arbol hoja = new Arbol();

hoja.info = elemento;

if (raiz == null)

{

raiz = hoja;

}

else

{

Arbol temp = raiz;

while (bandera != 1)

{

if (hoja.info < temp.info)

{

if (temp.izq == null)

{

temp.izq = hoja;

bandera = 1;

}

else

{

temp = temp.izq;

}

}

else

{

if (temp.der == null)

{

temp.der = hoja;

bandera = 1;

}

else

{

temp = temp.der;

}

}

}

}

}

public void preorden(Arbol temp)

{

if (temp != null)

{

Console.WriteLine(temp.info);

if (temp.izq != null)

{

preorden(temp.izq);

}

if (temp.der != null)

{

preorden(temp.der);

}

}

else

{

Console.WriteLine("El arbol binario esta vacio");

}

}

public void inorden(Arbol temp)

{

if (temp != null)

{

if (temp.izq != null)

{

inorden(temp.izq);

}

Console.WriteLine(temp.info);

if (temp.der != null)

{

inorden(temp.der);

}

}

else

{

Console.WriteLine("El arbol binario esta vacio");

}

}

public void posorden(Arbol temp)

{

if (temp != null)

{

if (temp.izq != null)

{

posorden(temp.izq);

}

if (temp.der != null)

{

posorden(temp.der);

}

Console.WriteLine(temp.info);

}

else { Console.WriteLine("El arbol binario esta vacio"); }

}

public void busquedarecursiva(Arbol temp, int key)

{

if (temp == null)

{

Console.WriteLine("El numero de control " + key + " no esta en el arbol binario");

}

else

{

if (key == temp.info)

{

Console.WriteLine("El numero de control " + key + " si esta en el arbol binario");

}

else

{

if (key < temp.info)

{

busquedarecursiva(temp.izq, key);

}

else

{

busquedarecursiva(temp.der, key);

}

}

}

}

public void busquedaiterativa(Arbol temp, int key)

{

bool encontrado = false;

while (temp != null && encontrado == false)

{

if (key == temp.info)

{

encontrado = true;

}

else

{

if (key < temp.info)

{

temp = temp.izq;

}

else

{

temp = temp.der;

}

}

}

if (encontrado == false)

{

Console.WriteLine("El numero de control " + key + " no esta en el arbol binario");

}

else

{

Console.WriteLine("El numero de control " + key + " si esta en el arbol binario");

}

}

public void eliminar()

{

Arbol p, q, v, s, t;

bool encontrado = false;

int x;

p = raiz;

q = null;

if (p != null)

{

Console.Write("Ingrese numero de control que deseas eliminar del Arbol binario:");

x = int.Parse(Console.ReadLine());

while ((p != null) && (encontrado == false))

{

if (p.info == x)

{

encontrado = true;

Console.WriteLine("El numero de control " + p.info + " sera eliminado del arbol binario");

}

else

{

q = p;

if (x < p.info)

{

p = p.izq;

}

else

{

p = p.der;

}

}

}

if (encontrado == true)

{

if (p.izq == null)

{

v = p.der;

}

else

{

if (p.der == null)

{

v = p.izq;

}

else

{

t = p;

v = p.der;

s = v.izq;

while (s != null)

{

t = v;

v = s;

s = v.izq;

}

if (t != p)

{

t.izq = v.der;

v.der = p.der;

}

v.izq = p.izq;

}

}

if (q == null)

{

raiz = v;

}

else

{

if (p == q.izq)

{

q.izq = v;

}

else

{

q.der = v;

}

}

}

else

{

Console.WriteLine("El numero de control " + x + " no esta en el arbol binario");

}

}

else

{

Console.WriteLine("El arbol binairo esta vacio");

}

}

static void Main(string[] args)

{

Stopwatch st = new Stopwatch();

long TotalInicio = System.GC.GetTotalMemory(true);

st.Start();

Random Ran = new Random();

Arbol A = new Arbol();

char opc;

do

{

Console.Clear();

Console.WriteLine("MENU ARBOL NUMEROS DE CONTROL");

Console.WriteLine("A) Inserción de Nodos.");

Console.WriteLine("B) Recorrido en Preorden, Inorden, Posorden.");

Console.WriteLine("C) Búsquedas de forma Recursiva e Iterativa");

Console.WriteLine("D) Eliminación de Nodos");

Console.WriteLine("E) Salir del programa");

Console.Write("Ingrese la opcion deseada: ");

opc = Console.ReadKey().KeyChar;

switch (opc)

{

case 'A':

Console.WriteLine("\n");

int elemento = (Ran.Next(9999999, 100000000));

Console.WriteLine("El numero de control introducido es: {0}", elemento);

A.insertar(elemento);

Console.WriteLine("Enter para voler");

Console.ReadKey();

break;

case 'B':

Console.WriteLine("\n");

Console.WriteLine("Recorrido en Preorden: ");

A.preorden(A.raiz);

Console.WriteLine("Recorrido en Inorden: ");

A.inorden(A.raiz);

Console.WriteLine("Recorrido en Posorden: ");

A.posorden(A.raiz);

Console.WriteLine("Enter para voler");

Console.ReadKey();

break;

case 'C':

Console.WriteLine("\n");

Console.Write("Ingrese el numero de control que desea buscar:");

int key = int.Parse(Console.ReadLine());

Console.WriteLine("Resultado en Busqueda recursiva: ");

A.busquedarecursiva(A.raiz, key);

Console.WriteLine("");

Console.WriteLine("Resultado en Busqueda iterativa: ");

A.busquedaiterativa(A.raiz, key);

Console.WriteLine("Enter para voler");

Console.ReadKey();

break;

case 'D':

Console.WriteLine("\n");

A.eliminar();

Console.WriteLine("Enter para volver");

Console.ReadKey();

break;

case 'E':

Console.WriteLine("\nDESPLIEGUE DE COMPLEJIDADES");

long TotalFinal = System.GC.GetTotalMemory(true);

Console.WriteLine("\nMemoria total: {0} bytes", TotalFinal - TotalInicio);

st.Stop();

Console.WriteLine("\nTIEMPO TRANSCURRIDO EN LA EJECUCION DEL PROGRAMA");

Console.WriteLine($"Tiempo: {st.Elapsed.TotalSeconds} segundos");

Console.WriteLine("\nPresione cualquier tecla para terminar...");

Console.ReadKey();

break;

default:

Console.WriteLine("\nOpcion invalida, intente otra vez.");

Console.ReadKey();

break;

}

} while (opc != 'E');

}

}

}

}

B

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Diagnostics;

namespace CovarrubiasLuis\_U4

{

class Program

{

class Arbol

{

string info;

Arbol izq, der;

public Arbol()

{

info = "";

izq = null;

der = null;

}

public Arbol raiz = null;

~Arbol() => Console.WriteLine("Memoria del Arbol liberada");

public void insertar(string elemento)

{

int bandera = 0;

Arbol hoja = new Arbol();

hoja.info = elemento;

if (raiz == null)

{

raiz = hoja;

}

else

{

Arbol temp = raiz;

while (bandera != 1)

{

if (hoja.info.CompareTo(temp.info) < 0)

{

if (temp.izq == null)

{

temp.izq = hoja;

bandera = 1;

}

else

{

temp = temp.izq;

}

}

else

{

if (temp.der == null)

{

temp.der = hoja;

bandera = 1;

}

else

{

temp = temp.der;

}

}

}

}

}

public void preorden(Arbol temp)

{

if (temp != null)

{

Console.WriteLine(temp.info);

if (temp.izq != null)

{

preorden(temp.izq);

}

if (temp.der != null)

{

preorden(temp.der);

}

}

else

{

Console.WriteLine("El arbol binario esta vacio");

}

}

public void inorden(Arbol temp)

{

if (temp != null)

{

if (temp.izq != null)

{

inorden(temp.izq);

}

Console.WriteLine(temp.info);

if (temp.der != null)

{

inorden(temp.der);

}

}

else

{

Console.WriteLine("El arbol binario esta vacio");

}

}

public void posorden(Arbol temp)

{

if (temp != null)

{

if (temp.izq != null)

{

posorden(temp.izq);

}

if (temp.der != null)

{

posorden(temp.der);

}

Console.WriteLine(temp.info);

}

else { Console.WriteLine("El arbol binario esta vacio"); }

}

public void busquedarecursiva(Arbol temp, string key)

{

if (temp == null)

{

Console.WriteLine("El nombre " + key + " no esta en el arbol binario");

}

else

{

if (key == temp.info)

{

Console.WriteLine("El nombre " + key + " si esta en el arbol binario");

}

else

{

if (key.CompareTo(temp.info) < 0)

{

busquedarecursiva(temp.izq, key);

}

else

{

busquedarecursiva(temp.der, key);

}

}

}

}

public void busquedaiterativa(Arbol temp, string key)

{

bool encontrado = false;

while (temp != null && encontrado == false)

{

if (key == temp.info)

{

encontrado = true;

}

else

{

if (key.CompareTo(temp.info) < 0)

{

temp = temp.izq;

}

else

{

temp = temp.der;

}

}

}

if (encontrado == false)

{

Console.WriteLine("El nombre " + key + " no esta en el arbol binario");

}

else

{

Console.WriteLine("El nombre " + key + " si esta en el arbol binario");

}

}

public void eliminar()

{

Arbol p, q, v, s, t;

bool encontrado = false;

string x;

p = raiz;

q = null;

if (p != null)

{

Console.Write("Ingrese nombre que deseas eliminar del Arbol binario:");

x = Console.ReadLine();

while ((p != null) && (encontrado == false))

{

if (p.info == x)

{

encontrado = true;

Console.WriteLine("El nombre " + p.info + " sera eliminado del arbol binario");

}

else

{

q = p;

if (x.CompareTo(p.info) < 0)

{

p = p.izq;

}

else

{

p = p.der;

}

}

}

if (encontrado == true)

{

if (p.izq == null)

{

v = p.der;

}

else

{

if (p.der == null)

{

v = p.izq;

}

else

{

t = p;

v = p.der;

s = v.izq;

while (s != null)

{

t = v;

v = s;

s = v.izq;

}

if (t != p)

{

t.izq = v.der;

v.der = p.der;

}

v.izq = p.izq;

}

}

if (q == null)

{

raiz = v;

}

else

{

if (p == q.izq)

{

q.izq = v;

}

else

{

q.der = v;

}

}

}

else

{

Console.WriteLine("El nombre " + x + " no esta en el arbol binario");

}

}

else

{

Console.WriteLine("El arbol binairo esta vacio");

}

}

static void Main(string[] args)

{

Stopwatch st = new Stopwatch();

long TotalInicio = System.GC.GetTotalMemory(true);

st.Start();

Arbol A = new Arbol();

char opc;

do

{

Console.Clear();

Console.WriteLine("MENU ARBOL DE NOMBRES");

Console.WriteLine("A) Inserción de Nodos.");

Console.WriteLine("B) Recorrido en Preorden, Inorden, Posorden.");

Console.WriteLine("C) Búsquedas de forma Recursiva e Iterativa");

Console.WriteLine("D) Eliminación de Nodos");

Console.WriteLine("E) Salir del programa");

Console.Write("Ingrese la opcion deseada: ");

opc = Console.ReadKey().KeyChar;

switch (opc)

{

case 'A':

Console.WriteLine("\n");

A.insertar("MICKEY");

A.insertar("NEMO");

A.insertar("MINNIE");

A.insertar("DONALD");

A.insertar("DAISY");

A.insertar("GOODY");

A.insertar("PACO");

Console.WriteLine("Elementos insertados correctamente");

Console.WriteLine("Enter para voler");

Console.ReadKey();

break;

case 'B':

Console.WriteLine("\n");

Console.WriteLine("Recorrido en Preorden: ");

A.preorden(A.raiz);

Console.WriteLine("Recorrido en Inorden: ");

A.inorden(A.raiz);

Console.WriteLine("Recorrido en Posorden: ");

A.posorden(A.raiz);

Console.WriteLine("Enter para voler");

Console.ReadKey();

break;

case 'C':

Console.WriteLine("\n");

Console.Write("Ingrese el nombre que desea buscar:");

string key = Console.ReadLine();

Console.WriteLine("Resultado en Busqueda recursiva: ");

A.busquedarecursiva(A.raiz, key);

Console.WriteLine("");

Console.WriteLine("Resultado en Busqueda iterativa: ");

A.busquedaiterativa(A.raiz, key);

Console.WriteLine("Enter para voler");

Console.ReadKey();

break;

case 'D':

Console.WriteLine("\n");

A.eliminar();

Console.WriteLine("Enter para volver");

Console.ReadKey();

break;

case 'E':

Console.WriteLine("\nDESPLIEGUE DE COMPLEJIDADES");

long TotalFinal = System.GC.GetTotalMemory(true);

Console.WriteLine("\nMemoria total: {0} bytes", TotalFinal - TotalInicio);

st.Stop();

Console.WriteLine("\nTIEMPO TRANSCURRIDO EN LA EJECUCION DEL PROGRAMA");

Console.WriteLine($"Tiempo: {st.Elapsed.TotalSeconds} segundos");

Console.WriteLine("\nPresione cualquier tecla para terminar...");

Console.ReadKey();

break;

default:

Console.WriteLine("\nOpcion invalida, intente otra vez.");

Console.ReadKey();

break;

}

} while (opc != 'E');

}

}

}

}